

Creative Technology Solutions

Gabe Sequeira-Bacher 2022

# **CS 230 Project Software Design**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 05/18/2022 | Gabe Sequeira-Bacher | Version one |
| 1.1 | 07/01/2022 | Gabe Sequeira-Bacher | Version Two |
| 1.2 | 07/12/2022 | Gabe Sequeira-Bacher | Version Three |

## [Executive Summary](#_sbfa50wo7nsh)

Our client, The Gaming Room, wants to expand their user base and increase revenue by developing a game for web browser and base it around the existing game for the Android Store called Draw It or Lose It. Myself and the staff will be working on setting up the development environment. I propose that we borrow at least some of the functionality of the Android app and refactor the code to be compatible with web browser software.

## [Design Constraints](#_2et92p0)

One or more teams must be able to compete in games.

Players and teams must use names/identifiers unique to themselves.

Team must consist of multiple players.

Only one instance of the game can exist at any given time.

Program can run on a variety of platforms.

Will be coded using the Java programming language.

It’s important that this version at least somewhat resembles the game already available on android. A certain amount of development time should be allocated to refactoring the existing Android code to work for multiple platforms.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

## [Domain Model](#_8h2ehzxfam4o)

In this program, there is a single driver class that creates individual games with corresponding players and teams. The class that’s called GameService is primarily responsible for the creation of single instances of these. Since it is using the singleton pattern of software development, only one instance of GameService is allowed to exist in the computer’s memory at any given point in time.

This program uses multiple classes. A generic class Entity is referenced by three of the six other classes. These three classes—Player, Team, and Game--are said to **inherit** the id and name attributes from the super class Entity, as well as the mutator methods thereof. Teams can have Players added to them, and Games make use of one or more Teams. The UML shorthand 0..\* indicates that the classes may have as few as zero or as many as multiple attributes in common amongst themselves. SingletonTester class gets used to check if the Singleton pattern is being adhered to (i.e. checks to make sure only one instance of Game is in existence).

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac computers are relatively simple to code with. They also have some of the most secure software out-of-the-box. They may be expensive to work with, however, as they are only programmed to work with other Apple products. | Linux is a very secure operating system with a minimal risk of security threats to it. It does a lot of what Mac can do except it might cost less than using Mac for server-side development. Since the technology is open-source, it is preferred by developers. One thing about it is Linux is not quite as intuitive as its counterparts Mac and Windows. Since not as many Linux machines are in circulation, it's more difficult to find drivers for hosting such as the client is asking for. | Windows is maybe the most intuitive system for programming which offers a lot of customizability. It is also very compatible with different programs. This is the most prone to viruses, however, and a development team running windows should invest in anti-virus software. There is also the issue of expensiveness. | These are widespread and can reach a broad audience. These also catch viruses far less frequently than desktop machines. It’s not as common to upgrade mobile hardware than just to replace an old phone, however. |
| **Client Side** | Developers might have to already know about using mac hardware beforehand. An advantage is it can run applications for mac, windows and Linux OS’. Updates are less frequent, however. | This is the most niche OS and the expertise would have to be greater for this than for windows or mac. Aside from Mac, however, it is the most secure system to use. It’s the cheapest to use, but there is more room for error with Linux. | This would take less expertise to be able to use client-side. Viruses could become an issue if protection is not already put in place. This is also the best for gaming in general. Price range is like mac as well. | Anyone can basically develop a mobile apps, there are even apps now where you can make your own app. It can be very cost-effective. It may require some degree of expertise to develop between different mobile systems. In this case it is not cost-effective. |
| **Development Tools** | Development tools could consist of Javascript, CSS and HTML. It gives a greater range of control over the mac system via the console alone than windows PCs. IDEs might be Eclipse, Visual Studio suite and notepad++. | Linux is astonishingly similar to Mac in terms of development tools/IDEs. | Runs similar IDEs to Linux operating systems. But Windows has more tools available for it. | Programs would be developed in Swift for Android, and Objective C or Java for Apple. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: Most likely the most cost-effective and powerful OS for this use case is Windows, as it is simply the most streamlined for gaming. It won’t cost that much, either and there is a lot of support.
2. **Operating Systems Architectures**: Windows operating system makes use of both kernel and user modalities. Included with all windows systems are the graphical user interface and File management.
3. **Storage Management**: Windows allows you to manage your own files on your drive and the amount of space you can use. You can also choose where to save files to. The use of cloud computing software could cut back on cost and runtimes.
4. **Memory Management**: This system uses physical and virtual address spaces. There are ways to create more virtual address space to use, however. The program would run quicker if such methods were used.
5. **Distributed Systems and Networks**: It’s best to use a tool that can cover a range of OS systems. Unity is one-such game creation engine. It is also the most cost-effective. A server that mostly handles games would be ideal for a project such as this one.
6. **Security**: The Windows VPN can protect passwords and valuable information from being stolen. The virtual address system is private. There is also spyware-blocking technology built into the windows device. Other antivirus software can also be purchased for a moderate price.